On the Efficiency of Strategies for Subdividing Polynomial Triangular Surface Patches

Jean Gallier

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA

`jean@saul.cis.upenn.edu`

February 1, 2008

Abstract. In this paper, we investigate the efficiency of various strategies for subdividing polynomial triangular surface patches. We give a simple algorithm performing a regular subdivision in four calls to the standard de Casteljau algorithm (in its subdivision version). A naive version uses twelve calls. We also show that any method for obtaining a regular subdivision using the standard de Casteljau algorithm requires at least 4 calls. Thus, our method is optimal. We give another subdivision algorithm using only three calls to the de Casteljau algorithm. Instead of being regular, the subdivision pattern is diamond-like. Finally, we present a "spider-like" subdivision scheme producing six subtriangles in four calls to the de Casteljau algorithm.

1

1 Introduction

In this paper, we investigate the efficiency of various strategies for subdividing polynomial triangular surface patches. Subdivision methods based on a version of the de Casteljau algorithm splitting a control net into control subnets (see Farin [5]) were investigated by Goldman [11], Boehm and Farin [2], Böhm [4], and Seidel [16] (see also Boehm, Farin, and Kahman [3], and Filip [8]). However, except for Böhm [4], these papers are not particularly concerned with minimizing the number of calls to the standard de Casteljau algorithm. Furthermore, some of these papers (notably Goldman [11]) use a version of the de Casteljau algorithm computing a 5-dimensional simplex of polar values, which is more expensive than the standard 3-dimensional version. In this paper, we give a simple algorithm performing a regular subdivision in four calls to the standard de Casteljau algorithm (in its subdivision version). A naive version uses twelve calls. We also show that any method for obtaining a regular subdivision using the standard de Casteljau algorithm requires at least 4 calls. Thus, our method is optimal. We give another subdivision algorithm using only three calls to the de Casteljau algorithm. Instead of being regular, the subdivision pattern is diamond-like. Finally, we present a "spider-like" subdivision scheme producing six subtriangles in four calls to the de Casteljau algorithm. Some familiarity with affine spaces and affine maps is assumed. Details can be found in Farin [7], Berger [1], or Gallier [9].

2 The Polar Form Approach to Polynomial Triangular Surface Patches

The deep reason why polynomial triangular surface patches can be effectively handled in terms of control points is that multivariate polynomials arise from multiaffine symmetric maps (see Ramshaw [14], Farin [7, 6], Hoschek and Lasser [12], or Gallier [9]). Denoting the affine plane $\mathbb{R}^2$ as $\mathcal{P}$, traditionally, a *polynomial surface* in $\mathbb{R}^n$ is a function $F \colon \mathcal{P} \to \mathbb{R}^n$, defined such that

$$x_1 = F_1(u, v),$$
$$\ldots = \ldots$$
$$x_n = F_n(u, v),$$

for all $u, v \in \mathbb{R}$, where $F_1(U, V), \ldots, F_m(U, V)$ are polynomials in $\mathbb{R}[U, V]$. Given a natural number m, if each polynomial $F_i(U, V)$ has total degree $\leq m$, we say that F is a *polynomial surface of total degree m*. The *trace of the surface F* is the set $F(\mathcal{P})$.

Now, given a polynomial surface F of total degree m in some affine space $\mathcal{E}$ (typically $\mathbb{R}^3$), there is unique symmetric and multiaffine map $f \colon \mathcal{P}^m \to \mathcal{E}$ such that

$$F(u, v) = f(\underbrace{(u, v), \ldots, (u, v)}_{m}),$$

for all $(u, v) \in \mathcal{P}$. The symmetric and multiaffine map f associated with F is called the *polar form of F*.

The above result is not hard to prove. Using linearity, it is enough to deal with a single monomial. Given a monomial $U^h V^k$, with $h + k = d \leq m$, it is easily shown that the symmetric multiaffine f form corresponding to $U^h V^k$ is given

$$f((u_1, v_1), \ldots, (u_m, v_m)) = \frac{h! k! (m - (h + k))!}{m!} \sum_{\substack{I \cup J \subseteq \{1, \ldots, m\} \\ I \cap J = \emptyset \\ \text{card}(I) = h, \, \text{card}(J) = k}} \left(\prod_{i \in I} u_i \right) \left(\prod_{j \in J} v_j \right).$$

Recall that a map $f \colon \mathbb{R}^d \to \mathbb{R}^n$ is *affine* if

$$f((1 - \lambda)a + \lambda b) = (1 - \lambda)f(a) + \lambda f(b),$$

for all $a, b \in \mathbb{R}^d$, and all $\lambda \in \mathbb{R}$. A map $f \colon \underbrace{\mathbb{R}^d \times \cdots \times \mathbb{R}^d}_{m} \to \mathbb{R}^n$ is *multiaffine* if it is affine in each of its arguments. A map $f \colon \underbrace{\mathbb{R}^d \times \cdots \times \mathbb{R}^d}_{m} \to \mathbb{R}^n$ is *symmetric* if it does not depend on the order of its arguments, i.e., $f(a_{\pi(1)}, \ldots, a_{\pi(m)}) = f(a_1, \ldots, a_m)$, for all $a_1, \ldots, a_m$, and all permutations π.

As an example, consider the following surface known as Enneper's surface:

$$F_1(U, V) = U - \frac{U^3}{3} + UV^2$$

$$F_2(U, V) = V - \frac{V^3}{3} + U^2 V$$

$$F_3(U, V) = U^2 - V^2.$$

We get the polar forms

$$f_1((U_1, V_1), (U_2, V_2), (U_3, V_3)) = \frac{U_1 + U_2 + U_3}{3} - \frac{U_1 U_2 U_3}{3}$$
$$+ \frac{U_1 V_2 V_3 + U_2 V_1 V_3 + U_3 V_1 V_2}{3}$$

$$f_2((U_1, V_1), (U_2, V_2), (U_3, V_3)) = \frac{V_1 + V_2 + V_3}{3} - \frac{V_1 V_2 V_3}{3}$$
$$+ \frac{U_1 U_2 V_3 + U_1 U_3 V_2 + U_2 U_3 V_1}{3}$$

$$f_3((U_1, V_1), (U_2, V_2), (U_3, V_3)) = \frac{U_1 U_2 + U_1 U_3 + U_2 U_3}{3} - \frac{V_1 V_2 + V_1 V_3 + V_2 V_3}{3}.$$

Furthermore, it turns out that any symmetric multiaffine map $f \colon \mathcal{P}^m \to \mathcal{E}$ is uniquely determined by a family of $\frac{(m+1)(m+2)}{2}$ points (where $\mathcal{E}$ is any affine space, say $\mathbb{R}^n$). Let

$$\Delta_m = \{(i, j, k) \in \mathbb{N}^3 \mid i + j + k = m\}.$$

The following lemma is easily shown (see Ramshaw [14] or Gallier [9]).

3

Lemma 2.1 *Given an affine frame Δrst in the plane $\mathcal{P}$, given a family $(b_{i,j,k})_{(i,j,k)\in\Delta_m}$ of $\frac{(m+1)(m+2)}{2}$ points in $\mathcal{E}$, there is a unique surface $F\colon\mathcal{P}\to\mathcal{E}$ of total degree m, defined by a symmetric m-affine polar form $f\colon\mathcal{P}^m\to\mathcal{E}$, such that*

$$f(\underbrace{r,\ldots,r}_{i},\underbrace{s,\ldots,s}_{j},\underbrace{t,\ldots,t}_{k}) = b_{i,j,k}$$

for all $(i,j,k)\in\Delta_m$. Furthermore, f is given by the expression

$$f(a_1,\ldots,a_m) = \sum_{\substack{I\cup J\cup K=\{1,\ldots,m\} \\ I,J,K\ disjoint}} \left(\prod_{i\in I}\lambda_i\right)\left(\prod_{j\in J}\mu_j\right)\left(\prod_{k\in K}\nu_k\right) f(\underbrace{r,\ldots,r}_{\mathrm{card}(I)},\underbrace{s,\ldots,s}_{\mathrm{card}(J)},\underbrace{t,\ldots,t}_{\mathrm{card}(K)}),$$

where $a_i = \lambda_i r + \mu_i s + \nu_i t$, with $\lambda_i + \mu_i + \nu_i = 1$, and $1 \le i \le m$.

For example, with respect to the standard frame $\Delta rst = ((1,0,0),(0,1,0),(0,0,1))$, we obtain the following 10 control points for the Enneper surface:

$$
\begin{array}{cccc}
& & f(r,r,r) & \\
& & (\tfrac{2}{3},0,1) & \\[4pt]
& f(r,r,t) & & f(r,r,s) \\
& (\tfrac{2}{3},0,\tfrac{1}{3}) & & (\tfrac{2}{3},\tfrac{2}{3},\tfrac{1}{3}) \\[4pt]
f(r,t,t) & & f(r,s,t) & & f(r,s,s) \\
(\tfrac{1}{3},0,0) & & (\tfrac{1}{3},\tfrac{1}{3},0) & & (\tfrac{2}{3},\tfrac{2}{3},-\tfrac{1}{3}) \\[4pt]
f(t,t,t) & f(s,t,t) & & f(s,s,t) & & f(s,s,s) \\
(0,0,0) & (0,\tfrac{1}{3},0) & & (0,\tfrac{2}{3},-\tfrac{1}{3}) & & (0,\tfrac{2}{3},-1)
\end{array}
$$

A family $\mathcal{N} = (b_{i,j,k})_{(i,j,k)\in\Delta_m}$ of $\frac{(m+1)(m+2)}{2}$ points in $\mathcal{E}$ is called a *(triangular) control net,* or *Bézier net.* Note that the points in

$$\Delta_m = \{(i,\,j,\,k)\in\mathbb{N}^3 \mid i+j+k=m\},$$

can be thought of as a triangular grid of points in $\mathcal{P}$. For example, when $m=5$, we have the following grid of 21 points:

$$
\begin{array}{ccccccccccc}
& & & & & 500 & & & & & \\
& & & & 401 & & 410 & & & & \\
& & & 302 & & 311 & & 320 & & & \\
& & 203 & & 212 & & 221 & & 230 & & \\
& 104 & & 113 & & 122 & & 131 & & 140 & \\
005 & & 014 & & 023 & & 032 & & 041 & & 050
\end{array}
$$

We intentionally let i be the row index, starting from the left lower corner, and j be the column index, also starting from the left lower corner. The control net $\mathcal{N} = (b_{i,j,k})_{(i,j,k) \in \Delta_m}$ can be viewed as an image of the triangular grid Δ_m in the affine space $\mathcal{E}$. It follows from Lemma 2.1 that there is a bijection between polynomial surfaces of degree m and control nets $\mathcal{N} = (b_{i,j,k})_{(i,j,k) \in \Delta_m}$. It should also be noted that there are efficient methods for computing control nets from parametric definitions, but this will be published elsewhere.

In the next section, we review a beautiful algorithm to compute a point $F(a)$ on a surface patch using affine interpolation steps, the de Casteljau algorithm.

3 The de Casteljau Algorithm for Triangular Patches

In this section, we explain in detail how the de Casteljau algorithm can be used to subdivide a triangular patch into three subpatches. For more details, see Farin [7, 6], Hoschek and Lasser [12], Risler [15], or Gallier [9]. In the next section, we will use versions of this algorithm to obtain a triangulation of a surface patch using recursive subdivision.

Given an affine frame Δrst, given a triangular control net $\mathcal{N} = (b_{i,j,k})_{(i,j,k) \in \Delta_m}$, recall that in terms of the polar form $f \colon \mathcal{P}^m \to \mathcal{E}$ of the polynomial surface $F \colon \mathcal{P} \to \mathcal{E}$ defined by $\mathcal{N}$, for every $(i,j,k) \in \Delta_m$, we have

$$b_{i,j,k} = f(\underbrace{r, \ldots, r}_{i}, \underbrace{s, \ldots, s}_{j}, \underbrace{t, \ldots, t}_{k}).$$

Given $a = \lambda r + \mu s + \nu t$ in $\mathcal{P}$, where $\lambda + \mu + \nu = 1$, in order to compute $F(a) = f(a, \ldots, a)$, the computation builds a sort of tetrahedron consisting of $m+1$ layers. The base layer consists of the original control points in $\mathcal{N}$, which are also denoted as $(b^0_{i,j,k})_{(i,j,k) \in \Delta_m}$. The other layers are computed in m stages, where at stage l, $1 \leq l \leq m$, the points $(b^l_{i,j,k})_{(i,j,k) \in \Delta_{m-l}}$ are computed such that

$$b^l_{i,j,k} = \lambda b^{l-1}_{i+1,j,k} + \mu b^{l-1}_{i,j+1,k} + \nu b^{l-1}_{i,j,k+1}.$$

During the last stage, the single point $b^m_{0,0,0}$ is computed. An easy induction shows that

$$b^l_{i,j,k} = f(\underbrace{a, \ldots, a}_{l}, \underbrace{r, \ldots, r}_{i}, \underbrace{s, \ldots, s}_{j}, \underbrace{t, \ldots, t}_{k}),$$

where $(i,j,k) \in \Delta_{m-l}$, and thus, $F(a) = b^m_{0,0,0}$.

Assuming that a is not on one of the edges of Δrst, the crux of the subdivision method is that the three other faces of the tetrahedron of polar values $b^l_{i,j,k}$ besides the face corresponding to the original control net, yield three control nets

$$\mathcal{N}ast = (b^l_{0,j,k})_{(l,j,k) \in \in \Delta_m},$$

corresponding to the base triangle Δast,

$$\mathcal{N}rat = (b_{i,0,k}^{l})_{(i,l,k)\in\Delta_m},$$

corresponding to the base triangle Δrat, and

$$\mathcal{N}rsa = (b_{i,j,0}^{l})_{(i,j,l)\in\Delta_m},$$

corresponding to the base triangle Δrsa. If a belongs to one of the edges, say rs, then the triangle Δrsa is flat, i.e. Δrsa is not an afine frame, and the net $\mathcal{N}rsa$ does not define the surface, but instead a curve. However, in such cases, the degenerate net $\mathcal{N}rsa$ is not needed anyway.

From an implementation point of view, we found it convenient to assume that a triangular net $\mathcal{N} = (b_{i,j,k})_{(i,j,k)\in\Delta_m}$ is represented as the list consisting of the concatenation of the $m+1$ rows

$$b_{i,0,m-i},\ b_{i,1,m-i-1},\ \ldots,\ b_{i,m-i,0},$$

i.e.,

$$f(\underbrace{r,\ldots,r}_{i},\ \underbrace{t,\ldots,t}_{m-i}),\ f(\underbrace{r,\ldots,r}_{i},\ s,\ \underbrace{t,\ldots,t}_{m-i-1}),\ \ldots,\ f(\underbrace{r,\ldots,r}_{i},\ \underbrace{s,\ldots,s}_{m-i-1},\ t),\ f(\underbrace{r,\ldots,r}_{i},\ \underbrace{s,\ldots,s}_{m-i}),$$

where $0 \leq i \leq m$. As a triangle, the net $\mathcal{N}$ is listed (from top-down) as

$$f(\underbrace{t,\ldots,t}_{m})\quad f(\underbrace{t,\ldots,t}_{m-1},\ s)\quad \ldots\quad f(t,\ \underbrace{s,\ldots,s}_{m-1})\quad f(\underbrace{s,\ldots,s}_{m})$$
$$\ldots\qquad\qquad\qquad\qquad\ldots$$
$$\ldots$$
$$f(\underbrace{r,\ldots,r}_{m-1},\ t)\quad f(\underbrace{r,\ldots,r}_{m-1},\ s)$$
$$f(\underbrace{r,\ldots,r}_{m})$$

The main advantage of this representation is that we can view the net $\mathcal{N}$ as a two-dimensional array net, such that $net[i,j] = b_{i,j,k}$ (with $i + j + k = m$). In fact, only a triangular portion of this array is filled. This way of representing control nets fits well with the convention that the affine frame Δrst is represented as follows:
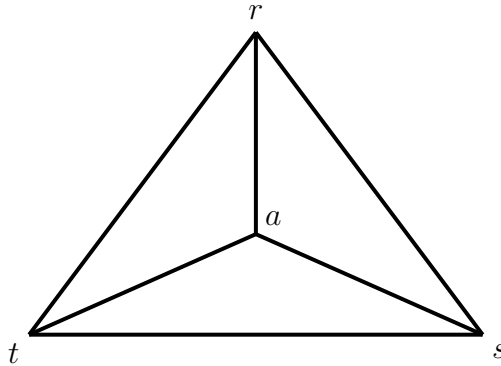
Figure 1: An affine frame

Instead of simply computing $F(a) = b^m_{0,0,0}$, the de Casteljau algorithm can be easily adapted to output the three nets $\mathcal{N}ast$, $\mathcal{N}rat$, and $\mathcal{N}rsa$. The function `sdecas3` does that. We also found it convenient to write three distinct functions `subdecas3ra`, `subdecas3sa`, and `subdecas3ta`, computing the control nets with respect to the affine frames Δast, Δart, and Δars. An implementation in *Mathematica* can be found in Gallier [9].

4   Regular Subdivision Of Triangular Patches

If we want to render a triangular surface patch F defined over the affine frame Δrst, it seems natural to subdivide Δrst into the three subtriangles Δars, Δast, and Δart, where $a = (1/3, 1/3, 1/3)$ is the center of gravity of the triangle Δrst, getting new control nets $\mathcal{N}ars$, $\mathcal{N}ast$ and $\mathcal{N}art$ using the functions described earlier, and repeat this process recursively. However, this process does not yield a good triangulation of the surface patch, because no progress is made on the edges rs, st, and tr, and thus, such a triangulation does not converge to the surface patch. Thus, in order to compute triangulations that converge to the surface patch, we need to subdivide the triangle Δrst in such a way that the edges of the affine frame are subdivided. There are many ways of performing such subdivisions, and we will propose a method which has the advantage of yielding a very regular triangulation, and of being very efficient. In fact, we give an optimal method for subdividing an affine frame using four calls to the standard de Casteljau algorithm in its subdivision version. A naive method would require twelve calls.

Goldman [11] proposed several subdivision algorithms, including one for splitting a triangular patch into four triangular subpatches, but his methods use a generalized version of the de Casteljau algorithm computing a 5-simplex of polar values. These methods are illustrated graphically in Boehm and Farin [2]. It should be noted that Boehm and Farin do mention that it is possible to compute the control net w.r.t. a new affine frame from the control net w.r.t. an original affine frame in three calls to the standard de Casteljau algorithm.

However, they do not explain how to split a triangular patch into four subpatches using four calls to the standard de Casteljau algorithm.

Goldman's subdivision methods can be justified in a very simple way as shown by Seidel [16]. Given a surface F of total degree m defined by a triangular control net $\mathcal{N} = (b_{i,j,k})_{(i,j,k) \in \Delta_m}$, w.r.t. the affine frame Δrst, for any n points $p_i = u_i r + v_i s + w_i t$ (where $u_i + v_i + w_i = 1$), the following $(n+2)$-simplex of points $b_{i,j,k}^{l_1,\ldots,l_n}$ where $i+j+k+l_1+\ldots+l_n = m$ is defined inductively as follows:

$$b_{i,j,k}^{0,\ldots,0} = b_{i,j,k},$$

$$b_{i,j,k}^{l_1,\ldots,l_h+1,\ldots,l_n} = u_h\, b_{i+1,j,k}^{l_1,\ldots,l_h,\ldots,l_n} + v_h\, b_{i,j+1,k}^{l_1,\ldots,l_h,\ldots,l_n} + w_h\, b_{i,j,k+1}^{l_1,\ldots,l_h,\ldots,l_n},$$

where $1 \leq h \leq n$.

If f is the polar form of F, it is easily shown that

$$b_{i,j,k}^{l_1,\ldots,l_n} = f(\underbrace{r,\ldots,r}_{i}, \underbrace{s,\ldots,s}_{j}, \underbrace{t,\ldots,t}_{k}, \underbrace{p_1,\ldots,p_1}_{l_1}, \ldots, \underbrace{p_n,\ldots,p_n}_{l_n}).$$

For $n = 0$, $F(p) = b_{0,0,0}^n$, as in the standard de Casteljau algorithm. For $n = 3$, $(b_{0,0,0}^{l_1,l_2,l_3})_{(l_1,l_2,l_2) \in \Delta_m}$ is a control net of F w.r.t. $\Delta p_1 p_2 p_3$.

In particular, if p_1, p_2, p_3 are chosen on the edges of Δrst, the subnets for the four subpatches are obtained. Goldman observes that some of the nets involved in the computation are trivial, but still, a 5-simplex of polar values is computed.

It was brought to our attention by Gerald Farin (and it is mentioned in Remark 2 of Seidel's paper [16], page 580) that Helmut Prautzsch showed in his dissertation (in German) [13] that regular subdivision into four subtriangles can be achieved in four calls to the standard de Casteljau algorithm. Prautzsch's method is briefly described in Böhm [4], page 348 (figure) and page 349 (in fact, with a typo, one of the barycentric coordinates listed as $\left(\frac{1}{2}, \frac{1}{2}, 1\right)$ should be $\left(\frac{1}{2}, \frac{1}{2}, 0\right)$). The order in which the four patches are obtained is slightly different from ours. Since Prautzsch's algorithm has not been discussed more extensively in the literature, we feel justified in presenting our method.

The subdivision strategy that we will follow is to divide the affine frame Δrst into four subtriangles Δabt, Δbac, Δcrb, and Δsca, where $a = (0, 1/2, 1/2)$, $b = (1/2, 0, 1/2)$, and $c = (1/2, 1/2, 0)$, are the middle points of the sides st, rt and rs respectively, as shown in the diagram below:
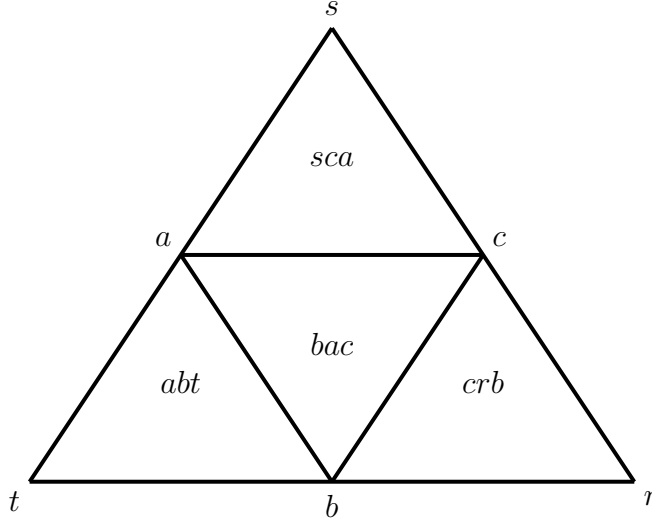
Figure 2: Subdividing an affine frame Δrst

First, we show that any method using the standard version of the de Casteljau algorithm for subdividing a triangular patch into 4 subpatches forming a regular pattern as above requires 4 calls. The crux of the argument is that a call to the de Casteljau algorithm in its subdivision version produces three subpatches containing only one new corner. We want to produce the four subpatches *abt*, *crb*, *sca*, and *abc*. After one subdivision step, we have three patches each involving exactly one of a, b, c. After two subdivision steps, we have six subpatches only two of which involve exactly two of a, b, c, since we can only subdivide a single patch, and since this patch only has one of a, b, c. Thus, at least three steps are needed to produce four subpatches involving at least two of a, b, c. If we produced the patch *abc* during the third subdivision step, we would have three patches involving exactly two of a, b, c, but the subdivision step that produced *abc* also produces two patches sharing the same vertex from (r, s, t). However, *abt*, *crb*, and *sca* do not share a vertex from (r, s, t). If *abc* was not produced during the third step, at least four steps are needed. Therefore, in all cases, at least four steps are needed to produce the required four subpatches.

We now present our algorithm. The first step is to compute the control net for the affine frame Δbat. This can be done using two steps. In the first step, split the triangle Δrst into the two triangles Δart and Δars, where $a = (0, 1/2, 1/2)$ is the middle of st. Using the function sdecas3 (with $a = (0, 1/2, 1/2)$), the nets $\mathcal{N}art$, $\mathcal{N}ast$, and $\mathcal{N}ars$ are obtained, and we throw away $\mathcal{N}ast$ (which is degenerate anyway). Then, we split Δart into the two triangles Δbat and Δbar. For this, we need the barycentric coordinates of b with respect to the triangle Δart, which turns out $(0, 1/2, 1/2)$. Using the function sdecas3, the nets $\mathcal{N}bat$, $\mathcal{N}brt$, and $\mathcal{N}bar$ are obtained, and we throw away $\mathcal{N}brt$.
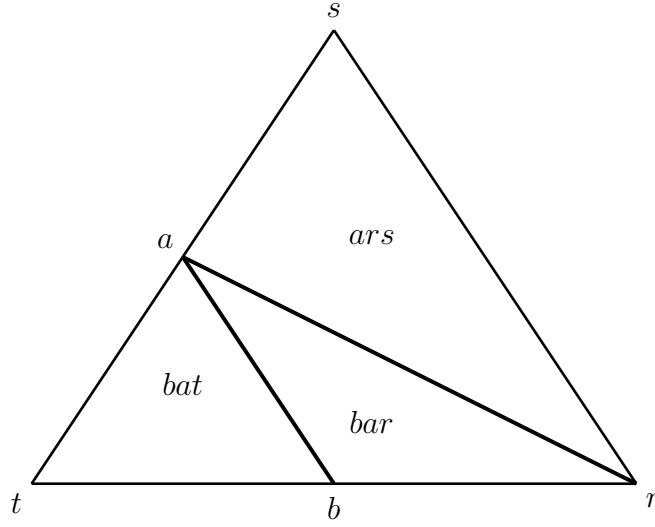
9

Figure 3: Computing the nets $\mathcal{N}bat$, $\mathcal{N}bar$ and $\mathcal{N}ars$ from $\mathcal{N}rst$

We will now compute the net $\mathcal{N}cas$ from the net $\mathcal{N}ars$. For this, we need the barycentric coordinates of c with respect to the triangle Δars, which turns out to be $(0, 1/2, 1/2)$. Using the function `subdecas3sa`, the net $\mathcal{N}cas$ is obtained.
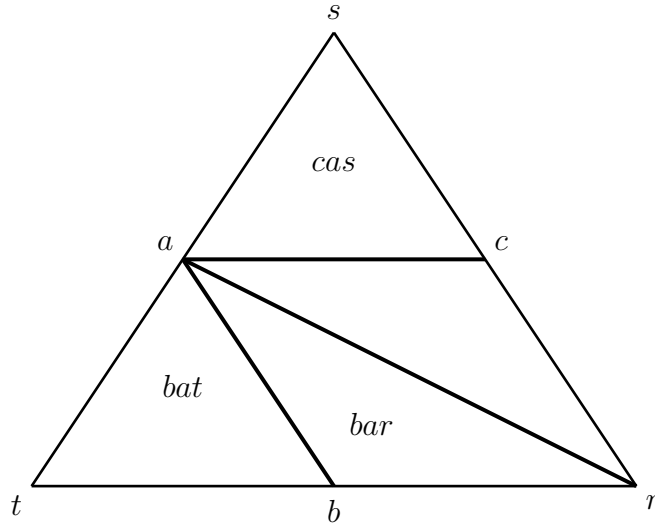

Figure 4: Computing the net $\mathcal{N}cas$ from $\mathcal{N}ars$

We can now compute the nets $\mathcal{N}cbr$ and $\mathcal{N}cba$ from the net $\mathcal{N}bar$. For this, we need the barycentric coordinates of c with respect to the affine frame Δbar which turns out to be

$(-1, 1, 1)$. Using the function `sdecas3`, the snet $\mathcal{N}cbr$, $\mathcal{N}car$, and $\mathcal{N}cba$ are obtained, and we throw away $\mathcal{N}car$.
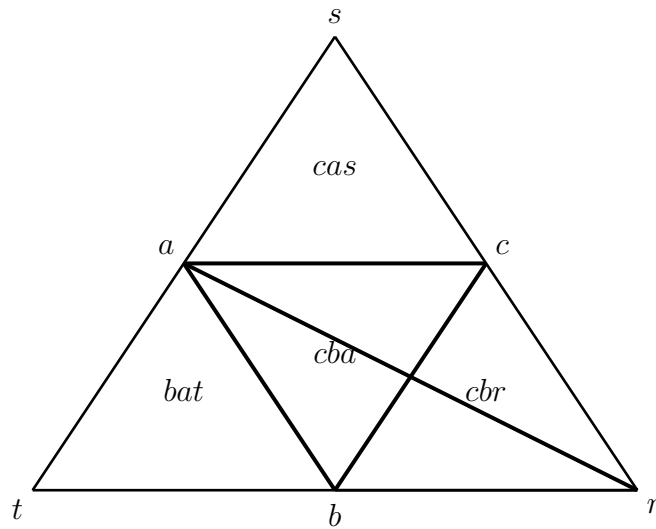

Figure 5: Computing the nets $\mathcal{N}cbr$ and $\mathcal{N}cba$ from $\mathcal{N}bar$

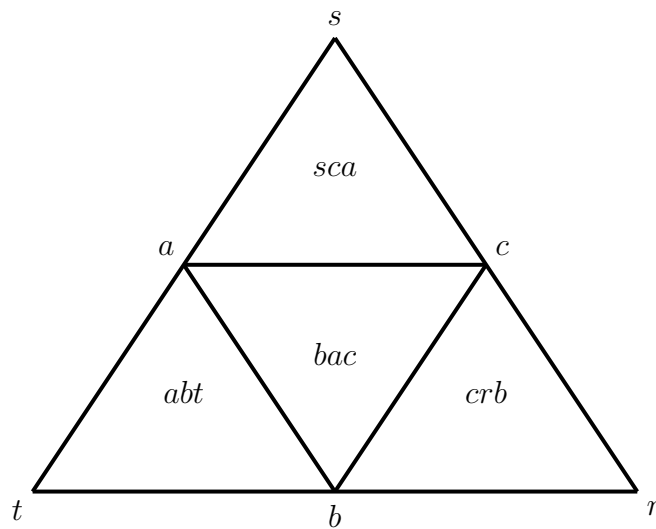Finally, we apply `transposej` to the net $\mathcal{N}bat$ to get the net $\mathcal{N}abt$, `transposek` to $\mathcal{N}cba$ to get the net $\mathcal{N}bac$, `transposej` followed by `transposek` to the net $\mathcal{N}cbr$ to get the net $\mathcal{N}crb$, and `transposek` twice to $\mathcal{N}cas$ to get the net $\mathcal{N}sca$,

Figure 6: Subdividing $\triangle rst$ into $\triangle abt$, $\triangle bac$, $\triangle crb$, and $\triangle sca$

11

Thus, using four calls to the de Casteljau algorithm, we obtained the nets $\mathcal{N}abt$, $\mathcal{N}bac$, $\mathcal{N}crb$, and $\mathcal{N}sca$.

Remarks:

(1) For debugging purposes, we assigned different colors to the patches corresponding to $\mathcal{N}abt$, $\mathcal{N}bac$, $\mathcal{N}crb$, and $\mathcal{N}sca$, and we found that they formed a particularly nice pattern under this ordering of the vertices of the triangles. In fact, $\mathcal{N}abt$ is blue, $\mathcal{N}bac$ is red, $\mathcal{N}crb$ is green, and $\mathcal{N}sca$ is yellow.

(2) In the last step of our algorithm, the subdivision step is performed with respect to a point of barycentric coordinates $(-1, 1, 1)$. One might worry that such a step involving a nonconvex combination is a source of numerical instability. We tested our algorithm on many different examples, and so far, without running into any problem. We also believe that such a nonconvex step is unavoidable if the standard de Casteljau algorithm (building a simplex of polar values of dimension 3) is used, but we are unable to prove this.

The subdivision algorithm just presented has been implemented in *Mathematica*, see Gallier [9]. The subdivision method is illustrated by the following example of a cubic patch specified by the control net

```
net = {{0, 0, 0}, {2, 0, 2}, {4, 0, 2}, {6, 0, 0},
       {1, 2, 2}, {3, 2, 5}, {5, 2, 2},
       {2, 4, 2}, {4, 4, 2},  {3, 6, 0}};
```
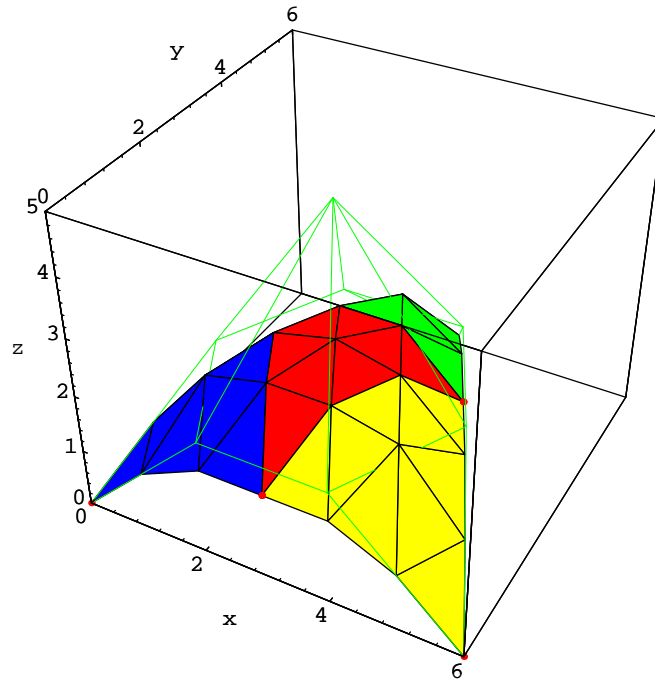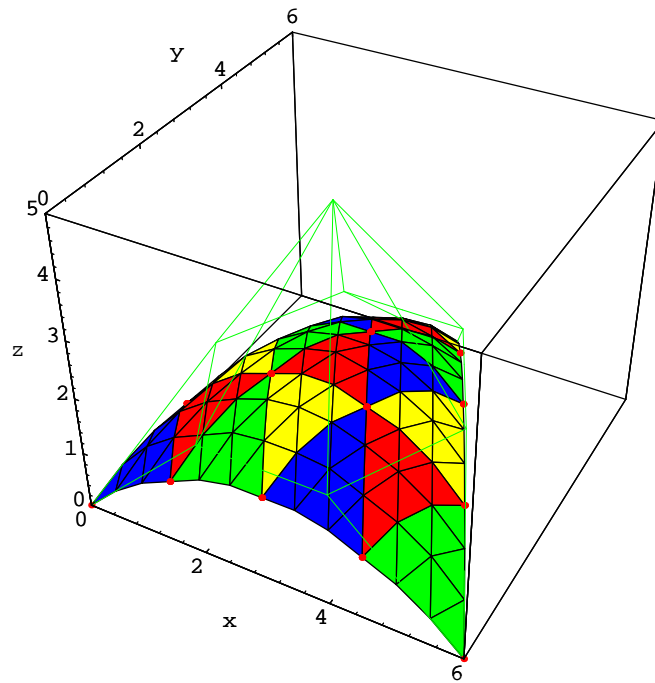
Figure 7: Subdivision, 1 iteration

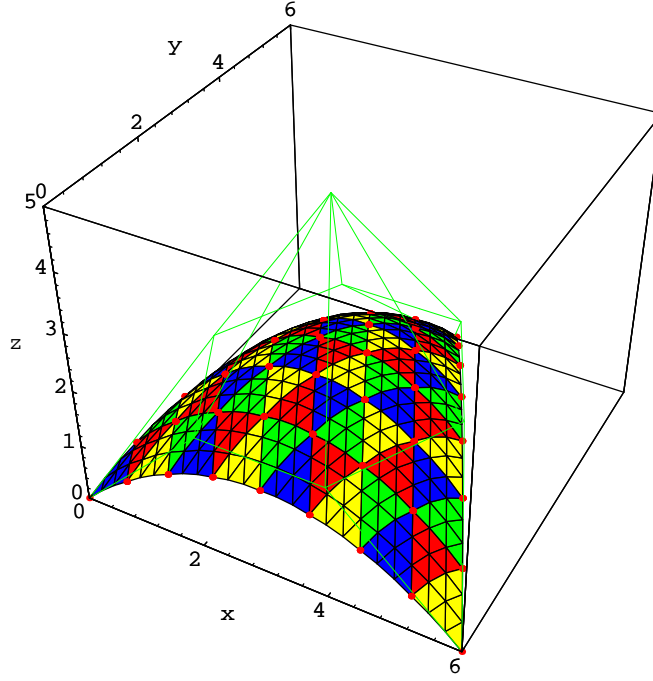Figure 8: Subdivision, 2 iterations

13

Figure 9: Subdivision, 3 iterations

After only three subdivision steps, the triangulation approximates the surface patch very well.

As another example of the use of the above functions, we can display a portion of a well known surface known as the "monkey saddle", defined by the equations

$$x = u, \quad y = v, \quad z = u^3 - 3uv^2.$$

Note that z is the real part of the complex number $(u + iv)^3$. It is easily shown that the monkey saddle is specified by the following triangular control net `monknet` over the standard affine frame Δrst, where $r = (1, 0, 0)$, $s = (0, 1, 0)$, and $t = (0, 0, 1)$.

```
monknet = {{0, 0, 0}, {0, 1/3, 0}, {0, 2/3, 0}, {0, 1, 0},
           {1/3, 0, 0}, {1/3, 1/3, 0},  {1/3, 2/3, -1},
           {2/3, 0, 0}, {2/3, 1/3, 0},  {1, 0, 1}};
```

We actually display the patch over the rectangle $[-1, 1] \times [-1, 1]$. This can be done by splitting the square into two triangles, and computing control nets with respect to these triangles. This is easy to do, and it is explained for example in Gallier [9]. Subdividing both nets 3 times, we get the following picture.
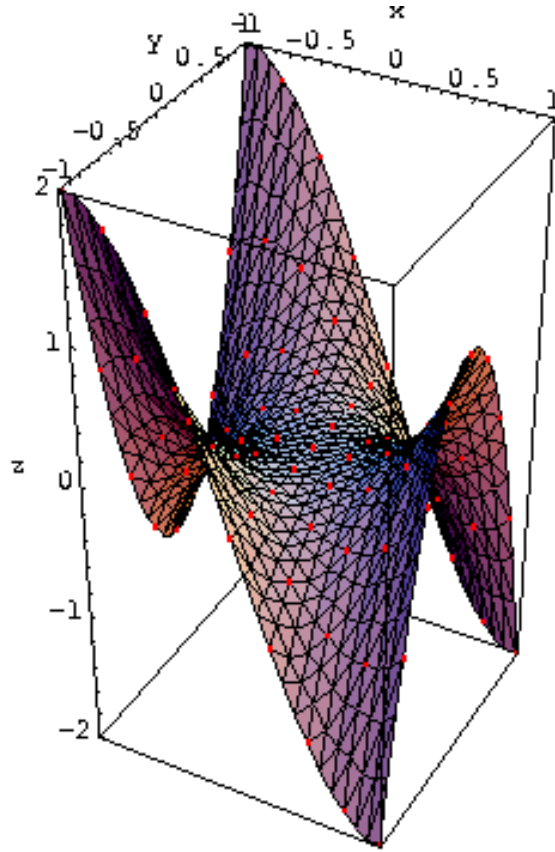
Figure 10: A monkey saddle, triangular subdivision

5   A Diamond-Shape Strategy For Subdivision

The strategy of the previous section was to split the affine frame Δrts into four congruent subtriangles. We were able to do this using four calls to the de Casteljau algorithm and we showed that it is not possible to do it in fewer calls.

However, it is possible to split the affine frame into four subtriangles using only three calls to the de Casteljau algorithm. The method consists in splitting the triangle Δrst into the four subtriangles Δbat, Δbar, Δcas, and Δcar:
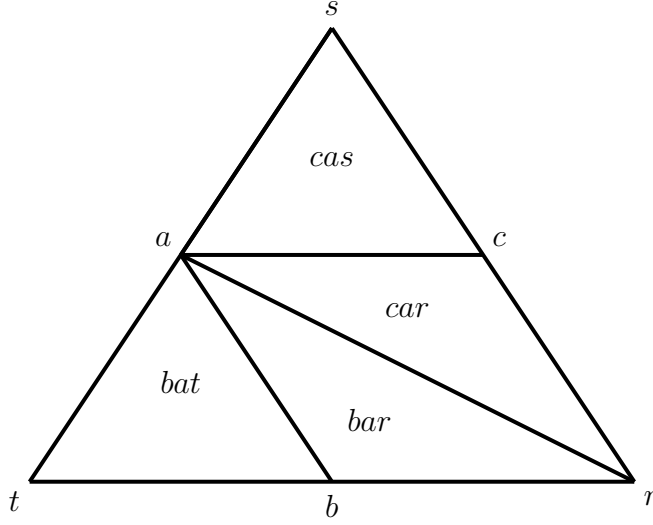
Figure 11: Diamond-style subdivision of an affine frame Δrst

This can be done by first computing the nets $\mathcal{N}art$ and $\mathcal{N}ars$, which can be done in one call to sdecas3 (dropping $\mathcal{N}ast$). Next, we split Δart into the two triangles Δbat and Δbar. For this, we need the barycentric coordinates of b with respect to the triangle Δart, which turns out $(0, 1/2, 1/2)$. Using the function sdecas3, the nets $\mathcal{N}bat$, $\mathcal{N}brt$, and $\mathcal{N}bar$ are obtained, and we throw away $\mathcal{N}brt$. Finally, we split Δars into the two triangles Δcas and Δcar. For this, we need the barycentric coordinates of c with respect to the triangle Δars, which turns out $(0, 1/2, 1/2)$. Using the function sdecas3, the nets $\mathcal{N}cas$, $\mathcal{N}crs$, and $\mathcal{N}car$ are obtained, and we throw away $\mathcal{N}crs$.

An implementation of the method is given in Gallier [9].

The result of subdividing two of three times reveals some diamond-shape subdividion patterns. For example, after three iterations, the dome surface is subdivided as follows:
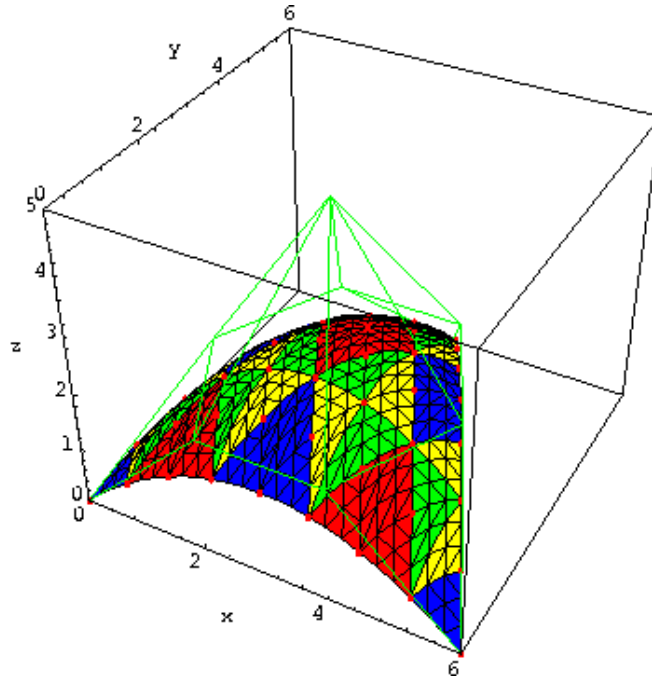
Figure 12: Diamond-style subdivision, 3 iterations

6   A Spider-Web Strategy For Subdivision

Is is also possible to split the affine frame into six subtriangles using only four calls to the de Casteljau algorithm.
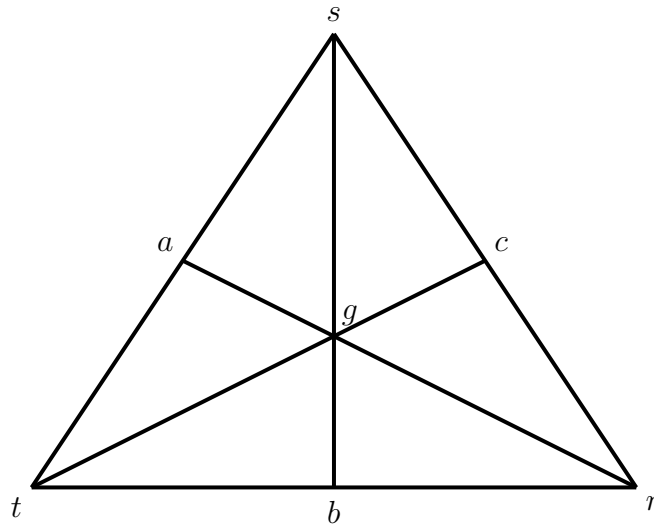

Figure 13: spider-web subdivision of an affine frame Δrst

The triangle Δrst is subdivided into the triangles Δbgt, Δbgr, Δagt, Δags, Δcgs, and Δcgr, as shown above, where g is the center of gravity. This can be done by first computing the nets $\mathcal{N}grt$, $\mathcal{N}gst$, and $\mathcal{N}grs$, which can be done in one call to sdecas3. We split Δgrt into the two triangles Δbgt and Δbgr using sdecas3 (throwing away Δbrt). We split Δgst into the two triangles Δagt and Δags using sdecas3 (throwing away Δast). Finally, we split Δgrs into the two triangles Δcgs and Δcgs using sdecas3 (throwing away Δcrs).

The result of subdividing recursively yields spider-web like patterns. For example, after three iterations, the dome surface is subdivided as follows:
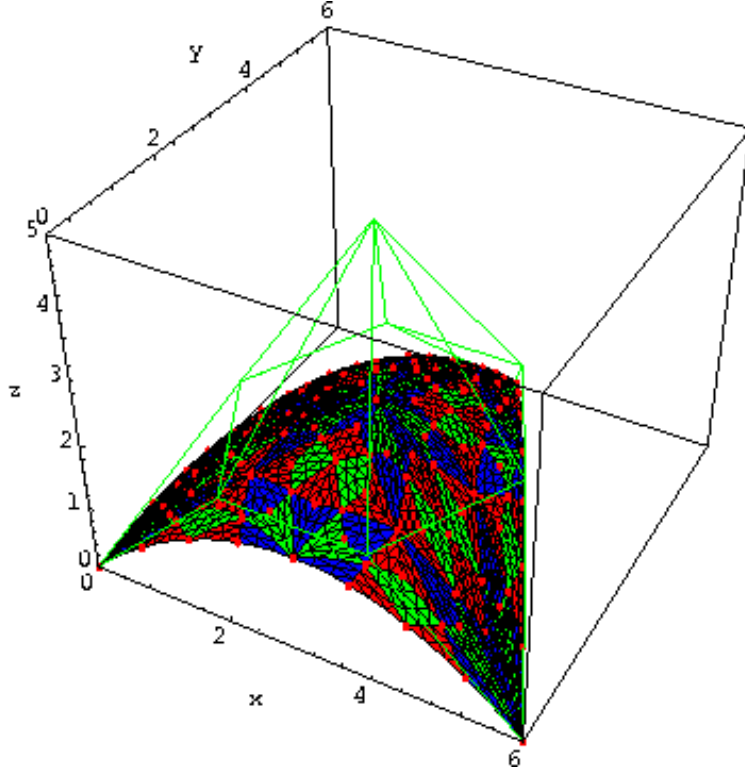

Figure 14: Spider-web style subdivision, 3 iterations

7 Conclusion

We have presented various strategies for subdividing polynomial triangular surface patches. We gave an algorithm performing a regular subdivision in four calls to the standard de Casteljau algorithm, and we showed that this method for obtaining a regular subdivision is optimal. We gave another subdivision algorithm using only three calls to the de Casteljau algorithm. Instead of being regular, the subdivision pattern is diamond-like. Finally, we presented a "spider-web" pattern subdivision scheme producing six subtriangles in four calls to the de Casteljau algorithm. These methods immediately apply to rational surface patches (Gallier [10]).

An amusing effect is obtained from the regular subdivision scheme if we omit the central triangle Δbac. We obtain a "fractalized" representation of the surface patch, in the sense that a Sierpinski gasket pattern is laid onto the patch! It would be interesting to investigate other subdivision strategies and the patterns that they induce, especially if the triangles are colored in various recursive manners.

References

[1] Marcel Berger. *Géométrie 1*. Nathan, 1990. English edition: Geometry 1, Universitext, Springer Verlag.

[2] W. Boehm and G. Farin. Letter to the editor. *Computer Aided Design*, 15(5):260–261, 1983. Concerning subdivison of Bézier triangles.

[3] W. Boehm, G. Farin, and J. Kahman. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1:1–60, 1984.

[4] Wolfgang Böhm. Subdividing multivariate splines. *Computer Aided Design*, 15:345–352, 1983.

[5] G. Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3:83–127, 1986.

[6] Gerald Farin. *NURB Curves and Surfaces, from Projective Geometry to practical use*. AK Peters, first edition, 1995.

[7] Gerald Farin. *Curves and Surfaces for CAGD*. Academic Press, fourth edition, 1998.

[8] D.J. Filip. Adapative subdivision algorithms for a set of Bézier triangles. *Computer Aided Design*, 18:74–78, 1986.

[9] Jean H. Gallier. *Curves and Surfaces In Geometric Modeling: Theory And Algorithms*. Morgan Kaufmann, first edition, 1999.

[10] Jean H. Gallier. Simple methods for drawing rational surfaces as four or six Bézier patches. Technical report, University of Pennsylvania, Levine Hall, 3330 Walnut, Philadelphia, PA 19104, 2006.

[11] R. N. Goldman. Subdivision algorithms for Bézier triangles. *Computer Aided Design*, 15:159–166, 1983.

[12] J. Hoschek and D. Lasser. *Computer Aided Geometric Design*. AK Peters, first edition, 1993.

[13] Helmut Prautzsch. *Unterteilungsalgorithmen für multivariate splines*. PhD thesis, TU Braunschweig, Braunschweig, Germany, 1984. Dissertation.

[14] Lyle Ramshaw. Blossoming: A connect-the-dots approach to splines. Technical report, Digital SRC, Palo Alto, CA 94301, 1987. Report No. 19.

[15] J.-J. Risler. *Mathematical Methods for CAD.* Masson, first edition, 1992.

[16] H.P. Seidel. A general subdivision theorem for Bézier triangles. In Tom Lyche and Larry Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 573–581. Academic Press, Boston, Mass, 1989.